FOUNDATION-0.3: ClickHouse Time-Series Schema - PART 2 (Execution & Testing)

🎯 CONTEXT

Phase: FOUNDATION (Week 1 - Day 2 Evening)

Component: ClickHouse Time-Series Database - Execution & Validation

Estimated Time: 10 min execution + 15 min testing

Complexity: MEDIUM

Risk Level: LOW

Files: Part 2 of 2 (Execution, testing, validation)

📦 PREREQUISITES

Must Have Completed:

- ✅ **PART 1** - All code files created
- ✅ **0.2a-0.2e** - PostgreSQL schemas complete

Verify PART 1 Files Exist:

```bash
cd ~/optiinfra

# Check all files created
ls -lh shared/clickhouse/migrations/init.sql
ls -lh shared/clickhouse/client.py
ls -lh shared/clickhouse/__init__.py
ls -lh shared/clickhouse/schemas/__init__.py
ls -lh shared/clickhouse/README.md

# Expected: All files present
```

🚀 STEP-BY-STEP EXECUTION

Step 1: Create Directory Structure

```bash
```

```
cd ~/optiinfra/shared

# Create clickhouse directory structure
mkdir -p clickhouse/schemas
mkdir -p clickhouse/migrations

# Verify structure
tree clickhouse/ -L 2
# Expected:
# clickhouse/
# ├── migrations/
# └── schemas/
```

Validation:

```
bash

# Check directories exist
[ -d "clickhouse/migrations" ] && echo "✅ migrations/ created" || echo "❌ migrations/ missing"
[ -d "clickhouse/schemas" ] && echo "✅ schemas/ created" || echo "❌ schemas/ missing"
```

Step 2: Create Initialization Script

```
bash
```

```
cd ~/optiinfra/shared/clickhouse/migrations

# Create init.sql (copy from PART 1, FILE 1)
cat > init.sql << 'EOF'
-- ============================================================================
-- OptiInfra ClickHouse Time-Series Database
-- Foundation Phase 0.3
-- ============================================================================

-- Create database
CREATE DATABASE IF NOT EXISTS optiinfra;
USE optiinfra;

[... COPY ENTIRE INIT.SQL FROM PART 1, FILE 1 ...]
EOF

# Verify file created
ls -lh init.sql
wc -l init.sql
# Expected: ~350 lines
```

Validation:

```
bash

# Check file size
FILE_SIZE=$(wc -l < init.sql)
if [ "$FILE_SIZE" -gt 300 ]; then
    echo "✅ init.sql created ($FILE_SIZE lines)"
else
    echo "❌ init.sql too small ($FILE_SIZE lines)"
fi

# Check for key tables
grep -q "cost_metrics_ts" init.sql && echo "✅ cost_metrics_ts found" || echo "❌ Missing"
grep -q "performance_metrics_ts" init.sql && echo "✅ performance_metrics_ts found" || echo "❌ Missing"
grep -q "resource_metrics_ts" init.sql && echo "✅ resource_metrics_ts found" || echo "❌ Missing"
grep -q "quality_metrics_ts" init.sql && echo "✅ quality_metrics_ts found" || echo "❌ Missing"
```

Step 3: Create Python Client

```bash
cd ~/optiinfra/shared/clickhouse

# Create client.py (copy from PART 1, FILE 2)
cat > client.py << 'EOF'
"""
ClickHouse client for high-frequency time-series metrics.
[... COPY ENTIRE CLIENT.PY FROM PART 1, FILE 2 ...]
"""
EOF

# Verify file
ls -lh client.py
wc -l client.py
# Expected: ~450 lines
```

Validation:

```bash
# Check file size
FILE_SIZE=$(wc -l < client.py)
if [ "$FILE_SIZE" -gt 400 ]; then
    echo "✅ client.py created ($FILE_SIZE lines)"
else
    echo "❌ client.py too small ($FILE_SIZE lines)"
fi

# Check for key classes
grep -q "class ClickHouseClient" client.py && echo "✅ ClickHouseClient class found" || echo "❌ Missing"
grep -q "def insert_cost_metrics" client.py && echo "✅ insert_cost_metrics found" || echo "❌ Missing"
grep -q "def query_cost_hourly" client.py && echo "✅ query_cost_hourly found" || echo "❌ Missing"
grep -q "get_clickhouse_client" client.py && echo "✅ get_clickhouse_client found" || echo "❌ Missing"
```

Step 4: Create Package Files

```bash
```

```
cd ~/optiinfra/shared/clickhouse

# Create __init__.py (copy from PART 1, FILE 3)
cat > __init__.py << 'EOF'
"""
ClickHouse time-series database package.
[... COPY FROM PART 1, FILE 3 ...]
"""
EOF

# Create schemas/__init__.py (copy from PART 1, FILE 4)
cat > schemas/__init__.py << 'EOF'
"""
ClickHouse schema initialization helpers.
"""

# Empty file for now, but can add schema utilities later
EOF

# Verify files
ls -lh __init__.py
ls -lh schemas/__init__.py
```

Validation:

```
bash

# Check files exist
[ -f "__init__.py" ] && echo "✅ __init__.py created" || echo "❌ Missing"
[ -f "schemas/__init__.py" ] && echo "✅ schemas/__init__.py created" || echo "❌ Missing"
```

Step 5: Update Requirements

```
bash
```

```
cd ~/optiinfra/shared

# Add clickhouse-driver to requirements.txt
echo "" >> requirements.txt
echo "# ClickHouse driver (FOUNDATION-0.3)" >> requirements.txt
echo "clickhouse-driver==0.2.6" >> requirements.txt

# Verify addition
tail -5 requirements.txt
```

Validation:

```
bash

# Check if clickhouse-driver is in requirements
grep -q "clickhouse-driver" requirements.txt && echo "✅ clickhouse-driver added" || echo "❌ Missing"
```

Step 6: Install Dependencies

```
bash

cd ~/optiinfra/shared

# Install clickhouse-driver
pip install clickhouse-driver==0.2.6

# Verify installation
python -c "import clickhouse_driver; print('✅ clickhouse-driver installed successfully')"
# Expected: ✅ clickhouse-driver installed successfully
```

Validation:

```
bash
```

```
# Test import
python << 'EOF'
try:
    import clickhouse_driver
    print("✅ clickhouse_driver imports successfully")
    print(f"   Version: {clickhouse_driver.__version__}")
except ImportError as e:
    print(f"❌ Import failed: {e}")
EOF
```

Step 7: Create README

bash

```
cd ~/optiinfra/shared/clickhouse

# Create README.md (copy from PART 1, FILE 6)
cat > README.md << 'EOF'
# ClickHouse Time-Series Database
[... COPY ENTIRE README FROM PART 1, FILE 6 ...]
EOF

# Verify file
ls -lh README.md
```

Validation:

bash

```
# Check README exists and has content
FILE_SIZE=$(wc -l < README.md)
if [ "$FILE_SIZE" -gt 50 ]; then
    echo "✅ README.md created ($FILE_SIZE lines)"
else
    echo "❌ README.md too small"
fi
```

Step 8: Verify ClickHouse is Running

bash

```
cd ~/optiinfra

# Check if ClickHouse container is running
docker ps | grep clickhouse

# If not running, start all services
docker-compose up -d

# Wait for ClickHouse to be ready (takes ~10 seconds)
echo "Waiting for ClickHouse to start..."
sleep 10

# Test connection
docker exec optiinfra-clickhouse clickhouse-client --query="SELECT 1"
# Expected output: 1
```

Validation:

```
bash
```

```
# Comprehensive ClickHouse check
echo "=== ClickHouse Health Check ==="

# 1. Container running?
if docker ps | grep -q clickhouse; then
    echo "✅ ClickHouse container is running"
else
    echo "❌ ClickHouse container is NOT running"
    echo "   Run: docker-compose up -d clickhouse"
    exit 1
fi

# 2. Can connect?
if docker exec optiinfra-clickhouse clickhouse-client --query="SELECT 1" > /dev/null 2>&1; then
    echo "✅ ClickHouse is responding"
else
    echo "❌ ClickHouse is not responding"
    echo "   Check logs: docker logs optiinfra-clickhouse"
    exit 1
fi

# 3. Check version
VERSION=$(docker exec optiinfra-clickhouse clickhouse-client --query="SELECT version()")
echo "✅ ClickHouse version: $VERSION"

echo "=== All checks passed! ==="
```

Step 9: Initialize ClickHouse Database

bash

```
cd ~/optiinfra

# Run initialization script
echo "Initializing ClickHouse database..."
docker exec -i optiinfra-clickhouse clickhouse-client < shared/clickhouse/migrations/init.sql

# Expected output:
# CREATE DATABASE IF NOT EXISTS optiinfra
# USE optiinfra
# CREATE TABLE IF NOT EXISTS cost_metrics_ts ...
# [etc for all 8 tables/views]

echo "✅ ClickHouse initialization complete"
```

Validation:

```
bash
```

```bash
# Check all tables were created
echo "=== Verifying ClickHouse Tables ==="

TABLES=$(docker exec optiinfra-clickhouse clickhouse-client --database=optiinfra --query="SHOW TABLES" | wc -l)

if [ "$TABLES" -eq 8 ]; then
    echo "✅ All 8 tables/views created"
else
    echo "❌ Expected 8 tables, found $TABLES"
    echo "   Tables created:"
    docker exec optiinfra-clickhouse clickhouse-client --database=optiinfra --query="SHOW TABLES"
    exit 1
fi

# List all tables
echo ""
echo "ClickHouse tables:"
docker exec optiinfra-clickhouse clickhouse-client --database=optiinfra --query="SHOW TABLES"

# Expected output:
# cost_metrics_hourly_mv
# cost_metrics_ts
# performance_metrics_hourly_mv
# performance_metrics_ts
# quality_metrics_hourly_mv
# quality_metrics_ts
# resource_metrics_hourly_mv
# resource_metrics_ts
```

Step 10: Verify Table Structures

bash

```
cd ~/optiinfra

# Check cost_metrics_ts structure
echo "=== cost_metrics_ts structure ==="
docker exec optiinfra-clickhouse clickhouse-client --database=optiinfra --query="DESCRIBE TABLE cost_metrics_ts"

# Check materialized view
echo ""
echo "=== cost_metrics_hourly_mv structure ==="
docker exec optiinfra-clickhouse clickhouse-client --database=optiinfra --query="DESCRIBE TABLE cost_metrics_hourly_m

# Check partitioning
echo ""
echo "=== Partition configuration ==="
docker exec optiinfra-clickhouse clickhouse-client --database=optiinfra --query="
SELECT
    table,
    partition_key,
    sorting_key
FROM system.tables
WHERE database = 'optiinfra' AND table LIKE '%_ts'
FORMAT Vertical
"
```

Validation:

bash

```
# Verify all base tables have TTL
echo "=== Checking TTL Configuration ==="

for table in cost_metrics_ts performance_metrics_ts resource_metrics_ts quality_metrics_ts; do
    TTL=$(docker exec optiinfra-clickhouse clickhouse-client --database=optiinfra --query="
        SELECT ttl_expression
        FROM system.tables
        WHERE database='optiinfra' AND table='$table'
    ")

    if [ -n "$TTL" ]; then
        echo "✅ $table has TTL: $TTL"
    else
        echo "❌ $table missing TTL"
    fi
done
```

Step 11: Test Python Client Connection

bash

```
cd ~/optiinfra

# Test client import and connection
python << 'EOF'
from shared.clickhouse.client import get_clickhouse_client

print("=== Testing ClickHouse Client ===")

# Get client
client = get_clickhouse_client()
print("✅ Client initialized")

# Test ping
if client.ping():
    print("✅ ClickHouse connection successful!")
else:
    print("❌ ClickHouse connection failed")
    exit(1)

# Show tables
result = client.client.execute("SHOW TABLES")
print(f"\n✅ Found {len(result)} tables:")
for table in result:
    print(f"  - {table[0]}")

print("\n=== All tests passed! ===")
EOF
```

Expected Output:

```
=== Testing ClickHouse Client ===
✅ Client initialized
✅ ClickHouse connection successful!

✅ Found 8 tables:
  - cost_metrics_hourly_mv
  - cost_metrics_ts
  - performance_metrics_hourly_mv
  - performance_metrics_ts
  - quality_metrics_hourly_mv
  - quality_metrics_ts
  - resource_metrics_hourly_mv
  - resource_metrics_ts

=== All tests passed! ===
```

Step 12: Test Data Insertion

bash

```
cd ~/optiinfra

# Insert test cost metrics
python << 'EOF'
from shared.clickhouse.client import get_clickhouse_client
from datetime import datetime
import uuid

print("=== Testing Data Insertion ===")

client = get_clickhouse_client()
test_customer_id = '123e4567-e89b-12d3-a456-426614174000'

# Insert 2 test cost metrics
cost_metrics = [
    {
        'timestamp': datetime.now(),
        'customer_id': test_customer_id,
        'cloud_provider': 'aws',
        'service_name': 'ec2',
        'instance_id': 'i-test123',
        'instance_type': 'm5.xlarge',
        'region': 'us-east-1',
        'cost_per_hour': 0.192,
        'utilization_percent': 45.5,
        'is_spot': 0,
        'is_reserved': 0
    },
    {
        'timestamp': datetime.now(),
        'customer_id': test_
```